Technical Report Machine Learning

Deep Learning using PyTorch

Disusun oleh:

Galih Karya Gemilang

TK-44-03

1103202098

Program Studi S1 Teknik Komputer

Fakultas Teknik Elektro

Universitas Telkom

2023

Daftar isi

1. Deep learning

Deep learning adalah salah satu subbidang dalam bidang kecerdasan buatan (artificial intelligence) yang berfokus pada pengembangan dan penggunaan algoritma-algoritma yang mirip dengan struktur dan fungsi jaringan saraf manusia. Tujuan utama dari deep learning adalah untuk mengembangkan sistem yang mampu belajar secara mandiri melalui pengalaman dan data, serta mampu melakukan tugas-tugas kompleks dengan kinerja yang semakin meningkat seiring waktu.

Deep learning menggunakan jaringan saraf tiruan yang terdiri dari banyak lapisan (layers) yang saling terhubung. Setiap lapisan ini terdiri dari unit-unit pemrosesan (neuron) yang menghitung dan mentransformasikan input menjadi output. Lapisan-lapisan tersebut membentuk hierarki pemrosesan informasi, di mana lapisan-lapisan awal belajar fitur-fitur sederhana dari data mentah, sedangkan lapisan-lapisan yang lebih dalam belajar fitur-fitur yang lebih kompleks.

Proses pembelajaran dalam deep learning dilakukan melalui dua tahap utama, yaitu tahap pelatihan (training) dan tahap pengujian (inference). Pada tahap pelatihan, deep learning menggunakan algoritma pembelajaran yang disebut backpropagation, yang memungkinkan jaringan saraf untuk menyesuaikan bobot-bobot (weights) dan bias-bias yang terkait dengan setiap neuron dalam jaringan. Pelatihan dilakukan dengan mempersembahkan sejumlah besar data latihan (training data) yang berisi contoh-contoh yang sudah diketahui keluarannya, sehingga jaringan saraf dapat belajar menyesuaikan diri dan menghasilkan keluaran yang benar. Setelah tahap pelatihan selesai, deep learning dapat digunakan untuk melakukan prediksi atau klasifikasi terhadap data baru pada tahap pengujian. Jaringan saraf yang telah terlatih akan menerima input baru dan menghasilkan output berdasarkan pola-pola yang telah dipelajari selama tahap pelatihan. Keunggulan deep learning terletak pada kemampuannya untuk mengenali pola-pola kompleks dan mampu belajar dari data yang tidak terstruktur.

Deep learning telah diterapkan dalam berbagai bidang, termasuk pengenalan wajah, deteksi objek, pengenalan suara, analisis teks, pengolahan bahasa alami, dan banyak lagi. Keberhasilan deep learning dalam beberapa tahun terakhir dapat dikaitkan dengan beberapa faktor, seperti kemajuan dalam komputasi yang memungkinkan pelatihan jaringan saraf yang lebih besar dan kompleks, serta ketersediaan data yang lebih besar untuk pelatihan.

Namun, deep learning juga memiliki beberapa tantangan. Salah satu tantangan utama adalah interpretabilitas model. Karena deep learning menggunakan jaringan saraf yang kompleks, sulit untuk memahami alasan di balik setiap keputusan yang diambil oleh model. Selain itu, deep learning juga membutuhkan sumber daya komputasi yang besar dan waktu pelatihan yang lama, terutama untuk jaringan yang sangat dalam dan data yang besar.

Secara keseluruhan, deep learning telah mengubah lanskap kecerdasan buatan dengan memberikan kemampuan yang luar biasa dalam memahami dan memproses data kompleks. Dengan terus berkembangnya teknologi dan penelitian di bidang ini, deep learning diharapkan dapat memberikan kontribusi yang signifikan dalam berbagai aspek kehidupan manusia di masa depan.

2. PyTorch

PyTorch adalah sebuah framework atau kerangka kerja (framework) sumber terbuka yang populer dalam bidang kecerdasan buatan (artificial intelligence) dan pengolahan data. Dikembangkan oleh Facebook AI Research (FAIR), PyTorch menawarkan alat dan fungsi yang kuat untuk membangun, melatih, dan menerapkan model jaringan saraf tiruan (neural networks) dalam berbagai aplikasi.

Salah satu keunggulan utama PyTorch adalah pendekatan yang intuitif dan fleksibel dalam pemrograman yang memungkinkan para peneliti dan praktisi di bidang kecerdasan buatan untuk dengan mudah mengembangkan model mereka. PyTorch menggunakan pendekatan yang disebut "Define-by-Run", di mana pengguna dapat secara dinamis mendefinisikan dan mengubah graf komputasi saat eksekusi program berlangsung.

Berikut adalah beberapa fitur utama dari PyTorch:

1. **Dinamika Komputasi**: PyTorch memungkinkan pengguna untuk melakukan operasi matematika dan komputasi tensor dengan cara yang mirip dengan numpy array. Tensor adalah struktur data utama dalam PyTorch dan memungkinkan pengguna untuk melakukan operasi seperti penjumlahan, perkalian, dan fungsi matematika lainnya.
2. **AutoGrad**: PyTorch menyediakan fungsionalitas otomatis diferensiasi (automatic differentiation) melalui modul AutoGrad. Dengan AutoGrad, PyTorch secara otomatis menghitung gradien (gradient) dari suatu fungsi terhadap parameter yang diperlukan dalam proses pelatihan model jaringan saraf. Ini memudahkan pelatihan model dan penggunaan algoritma pembelajaran berbasis gradien seperti backpropagation.
3. **Modul Jaringan Saraf**: PyTorch menyediakan berbagai modul yang dapat digunakan untuk membangun arsitektur jaringan saraf. Pengguna dapat dengan mudah mendefinisikan lapisan-lapisan, fungsi aktivasi, dan modul lainnya untuk membangun model yang kompleks. PyTorch juga menyediakan banyak modul jaringan saraf yang telah dibangun sebelumnya seperti Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), dan lain-lain.
4. **Penanganan GPU**: PyTorch memiliki dukungan yang kuat untuk komputasi GPU. Hal ini memungkinkan para pengguna untuk dengan mudah memanfaatkan kecepatan dan kekuatan komputasi paralel yang ditawarkan oleh GPU untuk melatih model mereka dengan cepat.

5. **Alat dan Utilitas**: PyTorch menyediakan berbagai alat dan utilitas untuk membantu pengguna dalam membangun, melatih, dan mengevaluasi model mereka. Ini termasuk fungsi untuk memuat dan mengolah data, alat visualisasi, fungsi pengukuran kinerja, dan banyak lagi.
6. **Integrasi dengan Ekosistem Python**: PyTorch diintegrasikan dengan baik dengan ekosistem Python yang luas. Pengguna dapat dengan mudah menggunakan pustaka Python lainnya seperti NumPy, SciPy, dan Pandas bersamaan dengan PyTorch untuk analisis data yang efisien.
7. **Komunitas yang Aktif**: PyTorch memiliki komunitas yang aktif dan berkembang pesat. Komunitas ini menyediakan sumber daya, tutorial, dan dukungan yang kaya bagi pengguna PyTorch. Juga, terdapat banyak model jaringan saraf yang telah dilatih sebelumnya yang tersedia secara gratis untuk digunakan atau diubah sesuai kebutuhan.

PyTorch telah digunakan secara luas dalam berbagai bidang, termasuk pengenalan gambar, pemrosesan bahasa alami, visi komputer, pengolahan suara, dan banyak lagi. Dengan kemampuannya yang kuat dan pendekatan yang intuitif, PyTorch terus menjadi salah satu pilihan utama bagi para peneliti dan praktisi di bidang kecerdasan buatan.

3. Persiapan data

Data yang digunakan para program yang telah dibuat berasal dari sumber yang berbeda-beda, tergantung penggunaan. Pada program yang pendek dan simpel, cenderung membuat dataset sendiri berupa angka untuk memberikan gambaran cara kerja programnya seperti apa menggunakan PyTorch.

Pada program, dataset yang digunakan ada yang berupa teks, yaitu wine dataset yang dibuat pada video di Youtube dengan tautan youtube.com/watch?v=c36lUUr864M&t=10378s dan breast cancer dataset yang berasal dari sklearn. Kemudian ada yang berupa gambar, yaitu CIFAR10 dan MNIST yang berasal dari PyTorch untuk pengujian algoritma CNN, TransferLearning, Tensorboard, dan Saving and Loading Models.

Data yang digunakan pada program juga sudah bersih sehingga tidak dilakukan data cleaning.

4. Pembuatan Model Deep Learning dengan PyTorch

Ambil salah satu program, yaitu program ‘13_CNN.py’. Model yang dibentuk pada algoritma CNN adalah sebagai berikut.

Lapisan pertama (conv1) merupakan lapisan konvolusi dengan parameter 3, 6, dan 5. Parameter pertama, yaitu 3, menunjukkan jumlah saluran input yang sesuai dengan jumlah kanal warna pada gambar (RGB). Parameter kedua, yaitu 6, menunjukkan jumlah saluran output atau fitur yang dihasilkan oleh lapisan konvolusi ini. Parameter terakhir, yaitu 5, merupakan ukuran kernel konvolusi.

Lapisan berikutnya adalah lapisan pooling (pool), yang menggunakan metode max pooling dengan ukuran jendela 2x2 dan langkah 2x2. Hal ini membantu mengurangi dimensi fitur dan mempertahankan fitur paling dominan dalam setiap jendela.

Selanjutnya, terdapat lapisan konvolusi kedua (conv2) dengan konfigurasi yang serupa dengan conv1, yaitu memiliki 6 saluran input dan 16 saluran output, dengan ukuran kernel 5x5.

Setelah lapisan konvolusi, terdapat tiga lapisan linear (fc1, fc2, dan fc3) yang bertanggung jawab untuk melakukan transformasi linear pada input. fc1 memiliki input fitur sebesar 16 * 5 * 5, dan output fitur sebesar 120. fc2 memiliki input fitur sebesar 120 dan output fitur sebesar 84. Sedangkan fc3 memiliki input fitur sebesar 84 dan output fitur sebesar 10, sesuai dengan jumlah kelas yang ingin diprediksi oleh model.

Dengan definisi ini, model ConvNet siap digunakan untuk melakukan proses pelatihan dan prediksi pada tugas-tugas pengolahan citra yang sesuai.

5. Pelatihan Model

Optimizer yang digunakan pada program adalah SDG. SGD (Stochastic Gradient Descent) adalah optimizer yang umum digunakan dalam pelatihan model deep learning. Algoritma ini menggunakan subset data pelatihan secara acak untuk menghitung gradien dan memperbarui parameter model. SGD membutuhkan pemilihan learning rate yang tepat untuk memastikan kecepatan konvergensi yang optimal. Selain itu, SGD dapat diperluas dengan metode seperti momentum, Nesterov accelerated gradient, atau optimizer adaptif lainnya. Meskipun sederhana, SGD tetap dapat menghasilkan model yang berkualitas jika diatur dengan baik.

Kemudian, loss function yang digunakan adalah CrossEntropyLoss. CrossEntropyLoss adalah fungsi loss yang digunakan dalam klasifikasi pada deep learning. Fungsi ini mengukur perbedaan antara probabilitas prediksi dan probabilitas target. CrossEntropyLoss digunakan bersama fungsi aktivasi Softmax dan membantu model untuk mempelajari hubungan antara fitur input dan label target dengan cara yang optimal.

6. Peningkatan Model

Hyperparameter adalah parameter yang digunakan untuk mengontrol perilaku dan kinerja model dalam proses pelatihan. Hyperparameter tidak dihasilkan secara otomatis oleh model, melainkan harus ditentukan secara manual oleh pengguna atau peneliti sebelum memulai pelatihan.

Hyperparameter mempengaruhi cara model belajar dan bagaimana parameter internalnya diperbarui selama pelatihan. Pilihan yang tepat untuk hyperparameter dapat mempengaruhi kecepatan konvergensi model, kemampuan generalisasi, dan performa akhir model.

Contoh umum hyperparameter dalam deep learning termasuk:

1. **Learning Rate**: Menentukan seberapa besar perubahan yang diperbarui pada parameter model setiap kali dilakukan pembaharuan. Learning rate yang terlalu tinggi dapat menyebabkan model tidak konvergen, sementara learning rate yang terlalu rendah dapat membuat pelatihan berjalan lambat. Learning rate yang digunakan pada program ini sebesar 0,001.
2. **Jumlah Epoch**: Menentukan berapa kali seluruh dataset pelatihan akan diperiksa oleh model. Jumlah epoch yang terlalu rendah dapat menyebabkan model tidak cukup belajar, sedangkan jumlah epoch yang terlalu tinggi dapat menyebabkan overfitting. Jumlah epoch yang digunakan pada program ini sebanyak 5
3. **Batch Size**: Menentukan berapa banyak sampel data yang akan digunakan pada setiap iterasi pelatihan. Ukuran batch yang terlalu kecil dapat menyebabkan model lambat konvergen, sedangkan ukuran batch yang terlalu besar dapat menghabiskan sumber daya komputasi yang besar. Ukuran batch yang digunakan pada program ini sebesar 4.

7. Evaluasi dan Validasi Model

Berdasarkan model yang sudah disusun, pemilihan optimizer dan loss function, dan konfigurasi dari hyper-parameternya, didapatkan akurasinya sebagai berikut.

```
Finished Training
Accuracy of the network: 48.55 %
Accuracy of plane: 31.2 %
Accuracy of car: 58.3 %
Accuracy of bird: 50.7 %
Accuracy of cat: 36.3 %
Accuracy of deer: 28.9 %
Accuracy of dog: 38.1 %
Accuracy of frog: 54.6 %
Accuracy of horse: 52.6 %
Accuracy of ship: 81.8 %
Accuracy of truck: 53.0 %
```

8. Penyimpanan Model

Saving dan loading model adalah proses yang penting dalam pengembangan dan produksi model deep learning, karena memungkinkan kita untuk menyimpan model yang telah dilatih untuk digunakan kembali di masa depan atau di lingkungan yang berbeda.

Pada program, opsi yang digunakan untuk menyimpan model adalah sebagai berikut.

```
PATH = './cnn.pth'
torch.save(model.state_dict(), PATH)
```

Kemudian file model yang sudah tersimpan, bisa dimuat ulang kembali untuk mempersingkat waktu karena Ketika melakukan modelling atau training akan membutuhkan waktu yang cukup lama. Untuk memuat model yang sudah tersimpan sebelumnya dapat dilakukan dengan cara salah satunya sebagai berikut.

```
loaded_model = Model(n_input_features=6)
loaded_model.load_state_dict(torch.load(FILE))
loaded_model.eval()
```

9. Kesimpulan

Dimulai dengan menjelaskan tentang dasar-dasar deep learning, termasuk arsitektur jaringan saraf, fungsi aktivasi, dan propagasi mundur (backpropagation). Saya kemudian memperkenalkan PyTorch sebagai framework yang populer untuk membangun dan melatih model deep learning, dengan fokus pada keunggulan dan fleksibilitasnya.

Berikutnya persiapan data, yang melibatkan langkah-langkah seperti pemuatan dataset, pemrosesan data, dan pembagian data menjadi set pelatihan, validasi, dan pengujian. Kami menyoroti transformasi data yang dapat diterapkan menggunakan PyTorch, seperti normalisasi atau augmentasi data.

Selanjutnya, saya menjelaskan tentang pembuatan model deep learning dengan PyTorch. Kami membahas penggunaan kelas nn.Module untuk mendefinisikan arsitektur model, termasuk lapisan-lapisan, parameter, dan fungsi aktivasi. Saya juga menunjukkan contoh kode yang mengilustrasikan pembuatan model convolutional neural network (CNN) menggunakan PyTorch.

Selanjutnya membahas tentang pelatihan model. Saya menggambarkan langkah-langkah dalam proses pelatihan, termasuk definisi fungsi loss, pemilihan optimizer seperti SGD. Kami juga membahas penggunaan hyper-parameter, terutama epoch untuk mencegah overfitting.

Di akhir laporan, kami melakukan evaluasi model yang telah dilatih dengan menggunakan metrik-metrik yang sesuai, seperti akurasi.